УДК 378.013

МОДЕЛИ И АЛГОРИТМЫ ГЕНЕРАЦИИ ЗАДАЧ В КОМПЬЮТЕРНОМ ТЕСТИРОВАНИИ

В.В. Кручинин, Ю.В. Морозова

Томский университет систем управления и радиоэлектроники
E-mail: kru@ie.tusur.ru

Рассматриваются модели и алгоритмы генераторов тестовых заданий и вопросов. Показана структура шаблона задачи и предложен обобщенный алгоритм работы генератора задачи на основе шаблона. Описана технология построения и использования шаблонов для генерации задач. Показаны конкретные примеры шаблонов генераторов задач и их реализация для различных дисциплин. Рассмотрены шаблоны генератора вопросов на основе алгоритмов. Описанные модели и алгоритмы реализованы в конкретных генераторах и внедрены в учебный процесс Томском межвузовском центре дистанционного образования.

Введение

Опыт создания и использования компьютерных контрольных работ и экзаменаторов, накопленный в Томском межвузовском центре дистанционного образования показал [1, 2]:

1) студенты быстро приспосабливаются к небольшому (100 вопросов) экзамену, заготавливают шпаргалки;
2) происходит простое механическое запоминание ответов на данный вопрос, поскольку в ответ нужно ввести конкретное число или выбрать конкретный вариант;
3) при одновременной сдаче экзамена в компьютерном классе вопросы для разных студентов могут быть одни и те же.

Таким образом, уменьшается эффективность проведения экзаменационных сессий и контрольных точек. Для устранения указанных недостатков предлагается несколько вариантов действий:

1) расширить количество вопросов в компьютерных экзаменационных и контрольных работах, используя имеющуюся технологию;
2) создать новую технологию, базирующуюся на идее использования "генераторов".

Первый вариант решения проблемы имеет ряд существенных недостатков:

1) преподаватель-методист должен записать новые вопросы, это всегда связано с большими затратами;
2) количество вопросов существенно не увеличится, через определенный достаточно небольшой период времени вопросы необходимо будет переписывать заново.

Второй вариант решения проблем основывается на идее создания и использования генератора тестовых заданий и вопросов.

Проблемы построения генераторов уже обсуждались в литературе [3–9]. Общие методы построения генераторов описаны в монографии [4]. Ниже предлагается развить модели и методы генерации задач.

1. Генерация задач

Шаблон – это эффективный инструмент символьных преобразований текста. Под шаблоном обычно понимают заготовку текста, в котором некоторые элементы можно изменять в соответствии с заданным алгоритмом. Шаблоны широко используются в программировании, например, шаблоны в C++, мощный и развитый механизм, на основе которого была развита и реализована идея генерирующего программирования [10].

Под шаблоном задачи будем понимать описание задачи, в котором исходные данные и/или часть задачи могут меняться. Рассмотрим эту идею на конкретном примере. Пусть имеется задача: ***У Пети было два яблока, а у Васи три. Сколько яблок было у Пети и Васи?***

Для того, чтобы сделать из этой задачи шаблон, необходимо вместо конкретных чисел поставить параметры и алгоритмы, генерирующие значения этих параметров. Тогда эта задача может быть записана как: ***У Пети было gen(x) яблока, а у Васи gen(y). Сколько яблок было у Пети и Васи?***

Здесь ***gen(x)*** и ***gen(y)*** – программа, генерирующая значения для переменной x и y, соответственно.

В тестовых системах наряду с формулировкой конкретной задачи необходимо иметь правильное решение задачи или правильный ответ. Поэтому к шаблону нужно приложить программу решения задачи по сгенерированным параметрам. Тогда шаблон задачи будет выглядеть следующим образом:

Правильный ответ (rez=solv(x,y)),

где ***solv(x,y)*** – программа вычисления правильного ответа.

При формулировке конкретного вопроса студенту программа случайно выбирает число для x, далее случайно выбирает число переменной y, вычисляет правильный ответ, подставляет полученные числа x и y в задачу и выводит эту конкретную задачу студенту.

Если параметр x может принять 20 различных значений, а параметр y – 30, то общее число вариантов задач такого класса будет 600. Это уже достаточно большая выборка.

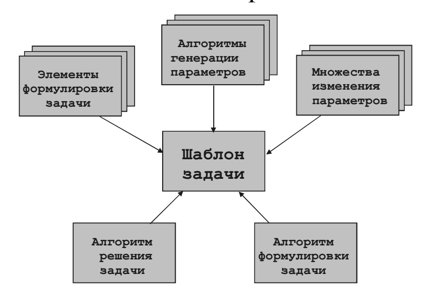


Рис. 1. *Структура шаблона задачи*

Для шаблона задачи необходимо (рис. 1):

1) выбрать некоторую задачу и выделить множество параметров, которое будет генерироваться;
2) построить алгоритм решения;
3) для каждого параметра записать множество изменения, это может быть список значений, интервалы или список интервалов;
4) для каждого параметра указать алгоритм генерации значения;
5) записать варианты формулировок задач. В некоторых случаях формулировка задачи может измениться в зависимости от значений параметров (в нашем примере: 11 яблок, но 2 яблока);
6) получить алгоритм формулировки задачи.

Тогда можно представить обобщенный алгоритм генерации задачи (рис. 2). Циклический характер алгоритма заключается в том, что при случайной генерации значений параметров задачи может быть ситуация, когда задача не имеет решения. Тогда процесс поиска значений параметров необходимо возобновить.

Рассмотрим некоторые примеры шаблонов задач для генератора заданий в компьютерном экзамене по математике [5].

Пример. Вычислить определитель 5-го порядка.

В общем случае для формулировки данной задачи необходимо сгенерировать соответствующую матрицу, дискриминант которой должен быть не равен нулю. Далее, используя алгоритм вычисления определителя, найти его значение. Использовать эту задачу для тестирования студентов авторы нашли нецелесообразным и предложили вариант матрицы специального вида:

$$\begin{vmatrix} 0 & b & c & a_1 & a_2 \\ 1 & x & y & z & u \\ 0 & b & c+1 & a_3 & a_4 \\ 0 & b & c+2 & a_5 & a_6 \\ 0 & b & c+3 & a_7 & a_8 \end{vmatrix}$$

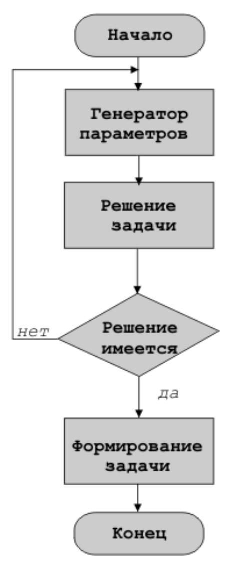


Рис. 2. Обобщенный алгоритм работы генератора задачи на основе шаблона

Для такого вида матрицы определитель считается по следующей формуле:

$$D = -b((a_5 + a_1 - 2a_3)(a_8 + a_4 - 2a_6) - \\ -(a_7 + a_3 - 2a_5)(a_6 + a_2 - 2a_4)), \quad (*)$$

где генерируются параметры ***b, c, x, y, z, u, {a}***, причем параметры ***c, x, y, z***, u могут быть любыми, ***b*** не равно нулю. Множество параметров генерируется из условия не равенство нулю определителя. Зная свойства разложения матрицы при вычислении определителя, сравнительно легко найти решение поставленной задачи. Тогда алгоритм генератора будет следующий:

Шаг 1. Генерируем любые значения для параметров ***c, x, y, z, u.***

Шаг 2. Генерируем значения для ***b*** и множества параметров ***{a}***.

Шаг 3. Вычисляем значение ***D*** по формуле (*).

Шаг 4. Если ***D*** равно нулю, то повторяем шаг 2.

Шаг 5. Формируем задание с полученными параметрами и записываем правильное решение.

Этот пример показывает, что алгоритм решения задачи может быть существенно упрощен. Количество вариантов заданий в этом примере является огромным числом, даже при таком упрощении. Например, если каждый значимый параметр будет иметь по 10 значений, то общее число вариантов может быть порядка 10^9.

Рассмотрим пример шаблона задачи из компьютерного экзамена по курсу "Магнитные элементы", разработанного профессором кафедры "Промышленная электроника" ТУСУРа В.П. Обрусником:

Определить плотность тока в обмотках магнитного элемента (усредненную) j (А/мм²) при K_{3K}=0,35 и параметрах катушек при:

$V_k \times 10^{-4}$, м³	ΔP_k, Вт	$\rho_к \times 10^{-8}$, Ом · м
p_1	p_2	p_3

Значение для параметра p_1 случайно выбирается из интервала {2,1;3,4}, для p_2={0,3;15} для p_3={10,1;45,5}.

Формула для расчета ответа:

$$j = \sqrt{\frac{\Delta P_k}{V_к \cdot \rho_к \cdot K_{3K}}}.$$

Пример шаблона задачи для контрольной работы по циклу "Цифровые и микропроцессорные устройства" [7]:

Оценить время (мкс) выполнения фрагмента программы микроконтроллером семейства МК51 при частоте кварца Z, МГц:

```
      MOV    R1, #M
M1:   MOV    R2, #N
      DJNZ   R2, $
      DJNZ   R1, M1
```

В условие задачи вставляются Z=12/K, M и N, причем переменные случайно выбираются из диапазона допустимых значений K=1,2,3,4,6,12; M=$\overline{(5{,}255)}$; N=$\overline{(5{,}255)}$. Ответ вычисляется по формуле X=K(1+3M+2MN). Нетрудно подсчитать, что генерируется 378 тысяч вариантов данного вопроса, отличающихся исходными значениями численных величин.

2. Генерация вопросов на основе алгоритмов

Алгоритмы являются универсальными инструментами описания деятельности в некоторой предметной области. Существует достаточно много различных способов представления алгоритмов: описание по шагам; различные блок-схемы; графовое преставление; описание алгоритмов в виде программ на некотором языке программирования и др.

Огромное количество алгоритмов имеет свойство цикличности: повторять некоторую последовательность действий (шагов), пока не наступит некоторое условие. Основываясь на этом свойстве, можно построить генератор вопросов. Для этого дадим ряд определений:

1. Телом цикла назовем последовательность шагов, которая должна многократно выполняться в цикле.
2. Итерацией цикла назовем однократное выполнение тела цикла.
3. Номер итерации цикла номер по порядку итерации.
4. Состояние итерации – значение всех переменных, связанных с данным циклом.
5. Начальное состояние – значения переменных цикла до первой итерации.
6. Условие завершения – условие, выполнение которого завершает работу цикла.

Рассмотрим простейший пример нахождения суммы натурального ряда:
Шаг 1: i=0, S=0,
Шаг 2: S=S+i, i=i+2,
Шаг 3: если i<n, то перейти на шаг 2.

Вопрос 1. Какое значение примет переменная *S* после завершения цикла, если *n=генерировать*();

Вопрос 2. Какое значение переменной *n* было установлено, если по завершению цикла значение *S =генерировать*().

Вопрос 3. Сколько итераций было выполнено, если по завершению цикла значение *S =генерировать*().

Вопрос 4. Какое значение примет переменная *S* на итерации *i=генерировать*().

Вопрос 5. На какой итерации значение *S* станет равным *k (k= генерировать*()).

Рассмотрим пример шаблона из тренажера по курсу "Алгоритмические языки и технология программирования" [11]:

1. Описание фрагмента программы:

```
char w4str[100];
void Wfunc3(char *b, char *e) {
      while( b!=e ) printf("%c",*b++);
      printf(" ");
}
void Wfunc4(char *b, char *e) {
char *stack=w4str;
      char *s=b;
      while( s!=e ) *stack++=*s++;
      for(s=b; s!=e; *s++=*--stack);
 Wfunc3(b, e);
}
void Word4(char *exp){
 char *beg, *end=exp;
 end=beg=Skip(end);
 while(*end) if( *end!=' ' ) end++; else break;
 if( end!=beg ) {
 Wfunc4( beg, end );
 Word4( end );
 }
}
```

2. Переменная *exp* является входной для данной программы, поэтому ее значение необходимо сгенерировать, т.е. для ее получения необходимо задать алгоритм генерации. Это можно сделать, задав список значений и записать алгоритм генерации, например, случайный.

3. Шаблоны для вопроса:

– Какое значение примет переменная *beg* при *i*-вызове функции *Wfunc4.*
– Что будет напечатано после *i*-го вызова функции *Word4.*
– Что будет напечатано после вызова функции *Word4*(*exp*).
– Сколько раз будет вызвана функция *Wfunc4.*
– Сколько раз будет вызвана функция *Wfunc3.*

Для каждого шаблона вопроса разрабатывается алгоритм получения правильного ответа.

Тогда алгоритм генерации вопроса будет следующий:

Шаг 1. Генерировать переменную *exp*;

Шаг 2. Выбрать вопрос, сгенерировать параметры (например, *i* в первом вопросе).

Шаг 3. Найти правильный ответ, используя соответствующий алгоритм решения.

Шаг 4. Сформировать тестовое задание, например:

"Дан следующий фрагмент программы: <фрагмент, описанный выше>

Переменная exp имеет значение "казак пилил сук". Что будет напечатано после вызова функции Word4(exp)?".

Правильный ответ: *казак лилип кус.*

Количество возможных вариантов зависит от генератора входной строки *exp* и генераторов параметров в вопросах. Опыт разработки таких генераторов показывает [11], что количество вариантов заданий может быть порядка 10^{16}. Это гарантирует каждому студенту получать индивидуальную задачу.

Заключение

В настоящее время в Томском межвузовском центре дистанционного образования разработана оригинальная технология создания генераторов вопросов и тестовых заданий. Эта технология базируется на инструментальной системе "Фея-3", которая успешно эксплуатируется в течение трех лет. Создано и эксплуатируется 20 генераторов, состоящих из 926 шаблонов задач. Уже первый опыт применения генераторов тестовых задач показал их эффективность для организации промежуточного и итогового контроля знаний студентов.

СПИСОК ЛИТЕРАТУРЫ

1. Воронин А.И., Исакова О.Ю., Кручинин В.В. Проблемы создания и сопровождения базы компьютерных учебных программ в Томском межвузовском центре дистанционного образования // Организация дистанционного обучения в Томском межвузовском центре дистанционного образования // Современное образование: Система и практика обеспечения качества: Тез. докл. регион. конф. – Томск: ТУСУР, 2002. – С. 103.
2. Исакова О.Ю., Кручинин В.В. Основные направления совершенствования контроля знаний в ТМЦДО // Единая образовательная среда: проблемы и пути развития: Матер. II Всеросс. научно-практ. конф. – Томск: Изд-во Томск. гос. ун-та, 2003. – С. 175–177.
3. Кручинин В.В. Разработка компьютерных учебных программ. – Томск: Изд-во Томск. гос. ун-та, 1998. – 211 с.
4. Башмаков А.И., Башмаков И.А. Разработка компьютерных и обучающих систем. – М.: Инф.-изд. дом "Филинъ", 2003. – 616 с.

5. Кручинин В.В., Магазинников Л.И. О повышении качества контролирующих материалов // Современное образование: Системы и практика обеспечения качества: Матер. регион. научно-метод. конф. – Томск: Изд-во ТУСУР, 2002. – С. 48.
6. Кашкарева Н.В., Кручинин В.В., Лычковская Л.Е. Генератор тестовых заданий по дисциплине "Английский язык" // Современное образование: Интеграция учебы, науки, производства: Матер. регион. научно-метод. конф. – Томск: Изд-во ТУСУР, 2003. – С. 49–50.
7. Егоркина Ю.В., Кручинин В.В., Шарапов А.В. Пакет генераторов тестовых заданий по циклу "Цифровые микропроцессорные устройства" // Современное образование: Интеграция учебы, науки, производства: Матер. регион. научно-метод. конф. – Томск: Изд-во ТУСУР, 2003. – С. 86–87.
8. Амзараков М.Б. Автоматическая генерация вариантов педагогического теста. http://ito.bitpro.ru/1999/II/6/6140.html
9. Левинская М.А. Автоматизированная генерация заданий по математике для контроля знаний учащихся. http://ifets.ieee.org/russian/depository/v5_i4/html/3.html
10. Musser D.R. Generic Programming. http://www.cs.rpi.edu/~musser/gp/index.html
11. Молочко М.В., Тимченко С.В. Генераторы по информатике в компьютерных экзаменах // Современное образование: инновации и конкурентоспособность: Матер. регион. научно-техн. конф. – Томск: Изд-во ТУСУР, 2004. – С. 105.

УДК 612.821.11.35